ОДЕССКИЙ НАЦИОНАЛЬНЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ИНСТИТУТ КОМПЬЮТЕРНЫХ СИСТЕМ

КАФЕДРА «КОМПЬЮТЕРНЫЕ ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ И СЕТИ»

Лабораторная работа №7

по дисциплине

«Системное программирование»

тема

«Многопоточные приложения»

Выполнили: Човганский Б.Ю.

Жмыхов Д.С.

Группа: АЕ-144

Бригада: 5

Одесса 2017

**Цель работы:**

Разработать программу с использованием среды разработки Visual Studio или DevCpp. Получить практические навыки работы с потоками.

**Задание на лабораторную работу:**

Вариант:

5.

Средство синхронизации потоков: атомарная операция.

Тип приложения: оконное.

Файл с исходными данными: отдельный для каждого потока.

Файл с результатами работы: общий для всех потоков.

Задание:

А. Создать многопоточное приложение с использованием Windows API, выполняющее перемножение матриц, размером 100\*100. Количество рабочих потоков равно (номер\_бригады+5). Исходные данные для работы всех потоков программы содержатся в одном или нескольких текстовых файлах (см. варианты заданий). Каждый рабочий поток считывает из файла очередные две матрицы и перемножает их. Результаты работы записываются в один или несколько выходных файлов (см. варианты заданий).

Б. Обеспечить корректный одновременный доступ к файлам со стороны множества потоков.

В. Обеспечить возможность корректной отмены работы указанного пользователем потока.

Г. Реализовать однопоточное приложение, выполняющего те же вычисления. Сравнить производительность одно- и многопоточного приложений.

Д. Отключить средства синхронизации потоков. Сравнить результаты работы программ с включенной и отключенной синхронизацией.

**Код программы:**

// Lab7.cpp: определяет точку входа для приложения.

//

#include "stdafx.h"

#include "Lab7.h"

#include <fstream>

#include <time.h>

#define MAX\_LOADSTRING 100

#define MATRIX\_SIZE 100

using namespace std;

// Глобальные переменные:

HINSTANCE hInst; // текущий экземпляр

WCHAR szTitle[MAX\_LOADSTRING]; // Текст строки заголовка

WCHAR szWindowClass[MAX\_LOADSTRING]; // имя класса главного окна

long GlobalThreadNumber=0;

long GlobalResultArray[MATRIX\_SIZE][MATRIX\_SIZE][20] = {0};

unsigned int end\_time;

unsigned int start\_time;

long Stack = 0;

// Отправить объявления функций, включенных в этот модуль кода:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: разместите код здесь.

// Инициализация глобальных строк

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_LAB7, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_LAB7));

MSG msg;

// Цикл основного сообщения:

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int) msg.wParam;

}

DWORD WINAPI Thread(PVOID param)

{

long ThreadNumber = Stack;

InterlockedAdd(&Stack, 1);

int a[MATRIX\_SIZE][MATRIX\_SIZE], b[MATRIX\_SIZE][MATRIX\_SIZE];

char buff[100] = { '\0' };

itoa(ThreadNumber, buff, 10);

strcat(buff, ".txt");

ifstream file(buff);

int counter = 0;

while (counter < 20) {

for (int i = 0; i < MATRIX\_SIZE; i++)

for (int j = 0; j < MATRIX\_SIZE; j++) {

file >> a[i][j];

}

for (int i = 0; i < MATRIX\_SIZE; i++)

for (int j = 0; j < MATRIX\_SIZE; j++) {

file >> b[i][j];

}

for (int i = 0; i < MATRIX\_SIZE; i++)

for (int j = 0; j < MATRIX\_SIZE; j++) {

long S = 0;

for (int m = 0; m < MATRIX\_SIZE; m++) {

S += a[i][m] \* b[m][j];

}

InterlockedExchange(&GlobalResultArray[i][j][ThreadNumber + counter], S);

}

counter += 10;

}

file.close();

InterlockedAdd(&GlobalThreadNumber, 1);

if (GlobalThreadNumber == 10) {

ofstream file("end.txt");

for (int n = 0; n < 20; n++)

for (int i = 0; i < MATRIX\_SIZE; i++) {

for (int j = 0; j < MATRIX\_SIZE; j++) {

file << GlobalResultArray[i][j][n] << " ";

}

}

file.close();

end\_time = clock();

}

return 0;

}

void Undo(int where) {

if (GlobalThreadNumber == 10) {

ofstream file("end.txt");

for (int n = 0; n < 20; n++) {

if (n != where && n != where + 10) {

for (int i = 0; i < MATRIX\_SIZE; i++) {

for (int j = 0; j < MATRIX\_SIZE; j++) {

file << GlobalResultArray[i][j][n] << " ";

}

}

}

}

file.close();

}

}

//

// ФУНКЦИЯ: MyRegisterClass()

//

// НАЗНАЧЕНИЕ: регистрирует класс окна.

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_LAB7));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_LAB7);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

//

// ФУНКЦИЯ: InitInstance(HINSTANCE, int)

//

// НАЗНАЧЕНИЕ: сохраняет обработку экземпляра и создает главное окно.

//

// КОММЕНТАРИИ:

//

// В данной функции дескриптор экземпляра сохраняется в глобальной переменной, а также

// создается и выводится на экран главное окно программы.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // Сохранить дескриптор экземпляра в глобальной переменной

HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

//

// ФУНКЦИЯ: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// НАЗНАЧЕНИЕ: обрабатывает сообщения в главном окне.

//

// WM\_COMMAND — обработать меню приложения

// WM\_PAINT — отрисовать главное окно

// WM\_DESTROY — отправить сообщение о выходе и вернуться

//

//

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

unsigned int Start, End;

switch (message)

{

case WM\_CREATE:

{

srand(time(0));

for (int i = 0; i < 10; i++) {

char buff[100] = { '\0' };

itoa(i, buff, 10);

strcat(buff, ".txt");

ofstream file(buff);

for (int a = 0; a < MATRIX\_SIZE\*MATRIX\_SIZE\*2; a++) file << rand()%10 <<" ";

file.close();

}

start\_time = clock();

for (int i = 0; i < 10; i++) {

CreateThread(NULL, 0, Thread, NULL, 0, NULL);

}

}

break;

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// Разобрать выбор в меню:

switch (wmId)

{

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

case IDM\_ZERO: Undo(0); break;

case IDM\_ONE: Undo(1); break;

case IDM\_TWO: Undo(2); break;

case IDM\_THREE: Undo(3); break;

case IDM\_FOUR: Undo(4); break;

case IDM\_FIVE: Undo(5); break;

case IDM\_SIX: Undo(6); break;

case IDM\_SEVEN: Undo(7); break;

case IDM\_EIGHT: Undo(8); break;

case IDM\_NINE: Undo(9); break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

Start = clock();

ofstream file2("end2.txt");

int a[MATRIX\_SIZE][MATRIX\_SIZE], b[MATRIX\_SIZE][MATRIX\_SIZE];

for (int bi = 0; bi < 10; bi++) {

char buff[100] = { '\0' };

itoa(bi, buff, 10);

strcat(buff, ".txt");

ifstream file(buff);

int counter = 0;

while (counter < 20) {

for (int i = 0; i < MATRIX\_SIZE; i++)

for (int j = 0; j < MATRIX\_SIZE; j++) {

file >> a[i][j];

}

for (int i = 0; i < MATRIX\_SIZE; i++)

for (int j = 0; j < MATRIX\_SIZE; j++) {

file >> b[i][j];

}

for (int i = 0; i < MATRIX\_SIZE; i++)

for (int j = 0; j < MATRIX\_SIZE; j++) {

long S = 0;

for (int m = 0; m < MATRIX\_SIZE; m++) {

S += a[i][m] \* b[m][j];

}

file2 << S << " ";

}

counter += 10;

}

file.close();

}

file2.close();

End = clock();

TCHAR str[60], tmp[20], tmp2[20];

\_tcscpy(str, \_T("Время работы без потоков в мкс: "));

\_i64tot(End-Start, tmp, 10);

\_tcscat(str, tmp);

TextOut(hdc, 0, 0, str, \_tcslen(str));

\_tcscpy(str, \_T("Время работы с потоками в мкс: "));

\_i64tot(end\_time-start\_time, tmp2, 10);

\_tcscat(str, tmp2);

TextOut(hdc, 0, 20, str, \_tcslen(str));

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

**Работа программы:**



**Вывод:** потоки - отличная возможность создать многозадачность в Windows, но мы можем видеть что все же эта операция дает очень небольшой выигрыш во времени, но это возможно из-за неудачного средства синхронизации, кажется что в данной задаче больше бы подошла критическая секция.